4 Analyse gefundener BPMN-Prozesse

Die gefundenen BPMN-Prozesse wurden weiter nach verschiedenen Merkmalen untersucht. Dafür wurden die im Abschnitt 1.2.2 beschriebenen Funktionen aus den Klassen FileStatistics, RepositoryStatistics, CodeMaatFunktions und ConstraintsChecker verwendet und aus Ergebnissen Grafiken gebildet.

Die Ergebnisse der Analysen können in zwei Kategorien geteilt werden: 1) allgemeine Statistiken über gefundene Dateien und ihre Repositorys und 2) Statistiken über Korrektheit der BPMN-Prozesse mit XSD-Serialisierungsformat, die mithilfe des BPMNspectors gebildet wurden.

4.1 Allgemeine Statistiken

Wir haben die allgemeinen Statistiken wiederum in zwei Gruppen geteilt: Statistiken über GitHub-Repositorys und Statistiken über gefundene BPMN-Prozesse.

4.1.1 Statistiken über GitHub-Repositorys

Die Repositorys mit BPMN-Prozessen wurden an solche Merkmale, wie Topprogrammiersprachen, Jahre der Erstellung und der letzten Änderung und Standorte der Repositoryentwickler untersucht.

4.1.1.1 Topprogrammiersprachen

Die Abbildung 2.1 zeigt die Topprogrammiersprachen der GitHub-Repositorys, die BPMN-Prozesse enthalten. Aus diesem Diagramm kann man ablesen, dass jedes zweite Repository (51,2%) Java als Topprogrammiersprache hat und mehr als jedes dritte Repository (37%) - JavaScript. Diesen Sprachen mit einem großen Abstand folgen Beschreibungssprache HTML (7,1%) und andere Programmiersprachen (<2%).

Um festzustellen, ob es einige Unterschiede zwischen den oben genannten Statistiken über Topprogrammiersprachen der GitHub-Repositorys mit BPMN-Prozessen und Statistiken über Topprogrammiersprachen von allen GitHub-Repositorys gibt, schauen wir jetzt auf die Abbildung 2.2. Diese zeigt Topprogrammiersprachen von GitHub-Repositorys, erstellt zwischen 2008 und 2018. Es ist zu sehen, dass JavaScript, Java und Python zu den Topprogrammiersprachen des GitHub gehören. Der Vergleich der Abbildungen 2.1 und 2.2 zeigt uns, dass es einige Ähnlichkeiten

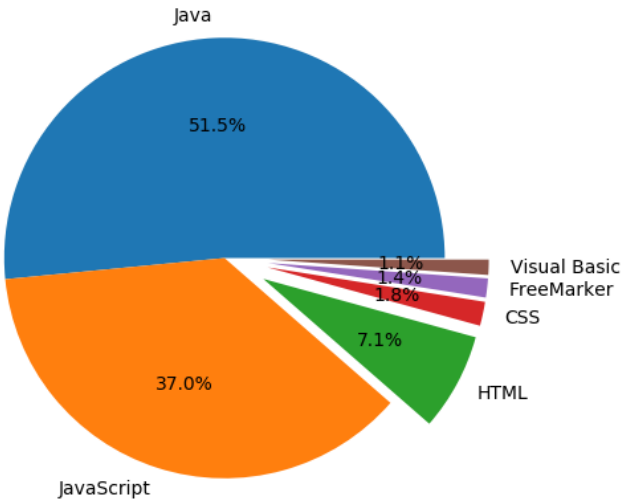

Abbildung 4.1: Dominante Programmiersprachen der Repositorys mit BPMN-Prozessen

und Unterschieden zwischen ihnen gibt. So gehören in beiden Abbildungen JavaScript und Java zu den zwei Topprogrammierapachen. Jedoch bei Repositorys mit BPMN-Prozessen ist Java an erster Stelle und bei allen GitHub-Repositorys - JavaScript. Auch alle anderen Beschreibung-und Programmiersprachen aus der Abbildung 2.1, wie HTML, CSS, FreeMarker und Visual Basic gehören nicht zu den Topprogrammiersprachen auf GitHub.

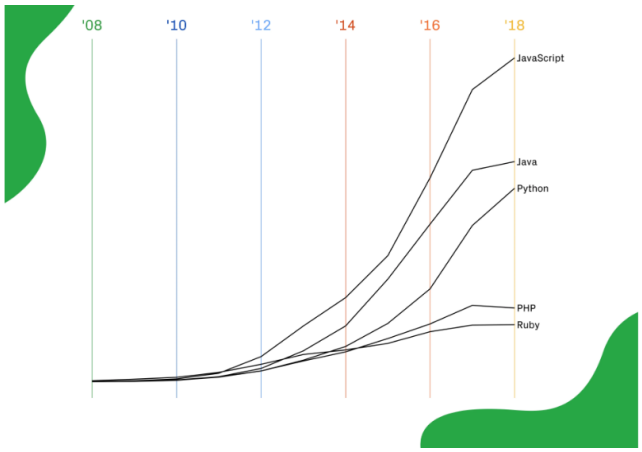

Abbildung 4.2: Topprogrammiersprachen von GitHub-Repositorys, erstellt zwischen 2008 und 2018 [4]

4.1.1.2 Erstellungsjahre und Jahre der letzten Änderung

Das Balkendiagramm auf der Abbildung 2.3 zeigt uns eine Dynamik der Erstellung von neuen GitHub-Repositorys mit BPMN-Prozessen. Es ist zu sehen, dass es zwischen 2005 und 2010 jährlich nicht mehr als 5 neue Repositorys mit BPMN-Prozessen erstellt wurden. Nach 2011 wurde diese Anzahl jedes Jahr deutlich hö-

her, als im Jahr davor. Der Grund dafür besteht vermutlich darin, dass im Januar 2011 die neue Version 2.0 von BPMN erschien. Seitdem wuchs die Anzahl der neu-erstellten Repositorys stetig und zum Beispiel wurden in 2017 fast 5-Mal so viele Repositorys mit BPMN-Prozessen erstellt, als drei Jahre davor in 2014. Im Durchschnitt betrug der jährliche Zuwachs der GitHub-Repositorys mit BPMN-Prozessen zwischen 2011 und 2017 ca. 71.7%. Die Daten für 2018 sind nicht komplett. Das ist darum, dass wir den GHTorren-Datensatz von 01.11.2018 benutzt haben, der Informationen nur bis diesem Datum enthielt.

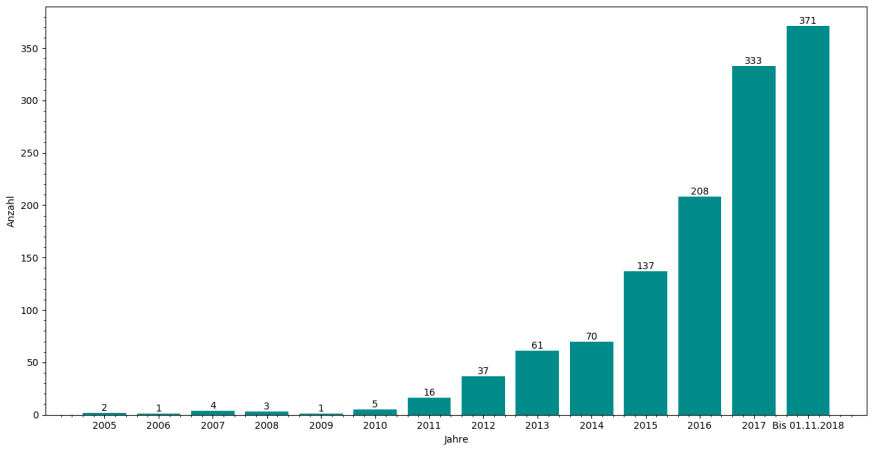

Abbildung 4.3: Jahre, in denen Repositorys mit BPMN-Prozessen erstellt wurden

Das Balkendiagramm in der Abbildung 2.4 liefert uns Informationen darüber, wann GitHub-Repositorys mit BPMN-Prozessen das letzte Mal geändert wurden. Dieses Diagramm zeigt, dass im Unterschied zur Abbildung 2.3, wo die ersten Repositorys im Jahre 2005 erstellt wurden, ersten Repositorys im Jahre 2010 abgeschlossen wurden. D.h. alle zwischen 2005 und 2010 erstellte Repositories waren mindestens bis 2010 aktiv. Aus dem Diagramm kann man auch entnehmen, dass die Anzahl der zum letzten Mal geänderten Repositorys jedes Jahr zwischen 2010 und 2018 immer größer war als im Jahr davor. Die Daten auf der X-Achse für das Jahr 2019 sind nicht komplett - nur bis 5. März. Das ist darum, dass das Klonen der GitHub-Repositorys am 05.03.2019 abgeschlossen wurde und wir Informationen über GitHub-Repositorys nur bis diesem Datum hatten.

Um die Tendenzen von Abbildungen 2.3 und 2.4 zu vergleichen, wurde aus beiden Grafiken ein gemeinsames Diagramm 2.5 erstellt und zwar für Jahre zwischen 2010 und 2017, für die es komplette Informationen auf beiden Grafiken gab. Das Diagramm 2.5 zeigt uns deutlich, dass die Anzahl der neu-erstellten Repositorys mit BPMN-Prozessen in einem der betrachteten Jahren größer war, als Anzahl der Repositorys, die in jeweiligem Jahr zum letzten Mal geändert wurden. Daraus kann man schließen, dass in diesen Jahren mehr öffentlichen GitHub-Repositorys mit BPMN-Prozessen erstellt, als abgeschlossen wurden.

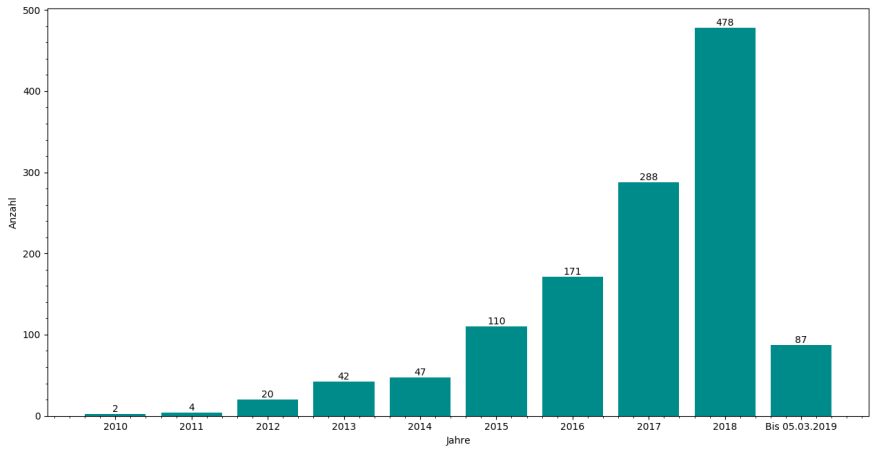

Abbildung 4.4: Jahre, in denen Repositorys mit BPMN-Prozessen letztes Mal geändert wurden

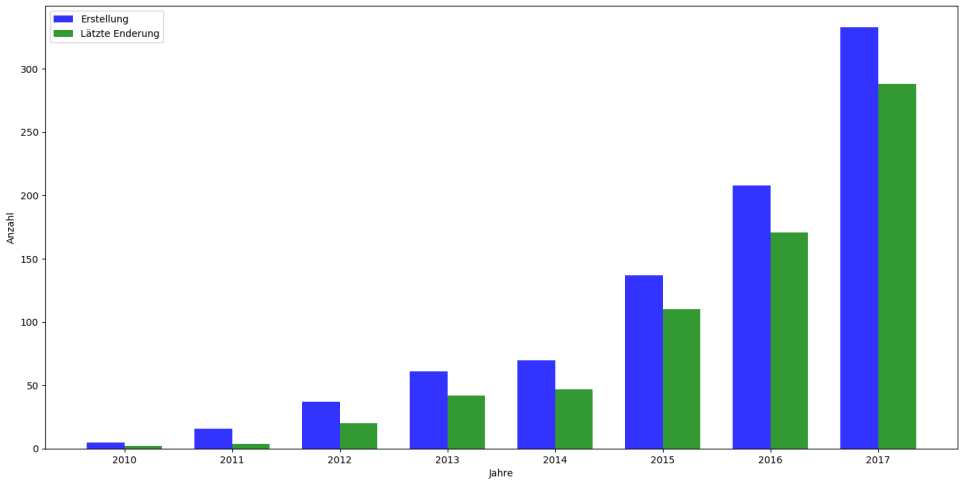

Abbildung 4.5: Jahre, in denen Repositorys mit BPMN-Prozessen erstellt und letztes Mal geändert wurden

4.1.1.3 Standorte der Repositorysentwickler

Das andere wichtige Merkmal über Repositorys, das untersucht wurde, war Standorte, wo sich die meisten Entwickler der Repositorys mit BPMN-Prozessen befinden. Die Ergebnisse sind auf der Abbildung 2.6 dargestellt. Diese Grafik enthält jedoch Standorte nur von ca. 31,6% von allen GitHub-Repositorys mit BPMN-Prozessen. Grund dafür besteht darin, dass die Datenbanktabelle „users“ (Abbildung 1.2), derer Inhalt aus dem Datensatz von GHTorrent-Projekt stammt und die Informationen über alle GitHub-Benutzer enthält, für viele von ihnen im Feld „country_code“ keine Angabe hat.

Bei Betrachten der Abbildung 2.6 fällt sofort auf, dass die meisten Entwickler, die zu den GitHub-Repositorys mit BPMN-Prozessen ihr Code schrieben, aus China (92), Deutschland (90) und aus den USA stammen (90). Ihnen mit einem großen Abstand folgen Programmierer aus der Schweiz (24), Frankreich (22) und anderen Ländern.

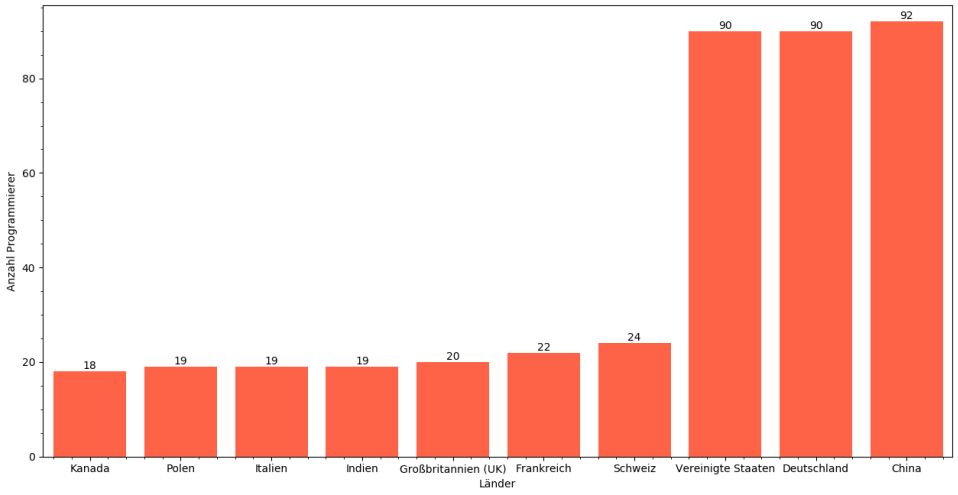

Abbildung 4.6: Standorte mit den meisten GitHub-Benutzer, die Code zu Repositorys mit BPMN-Prozessen schrieben

Die Abbildung 2.7 enthält Namen der Länder, aus denen die meisten GitHub-Benutzer kommen. Diese Abbildung ähnelt der Abbildung 2.6 in dem, dass auf beiden USA und China zu zweier Länder mit den meisten GitHub-Benutzer gehören. Aber im Gegensatz zur Abbildung 2.6, wo Deutschland und USA den zweiten Platzt teilen, befindet sich dieses europäische Land auf der Abbildung 2.7 auf dem 5. Platzt. Insgesamt kommen auf beiden Abbildungen sieben dieselben Länder vor.

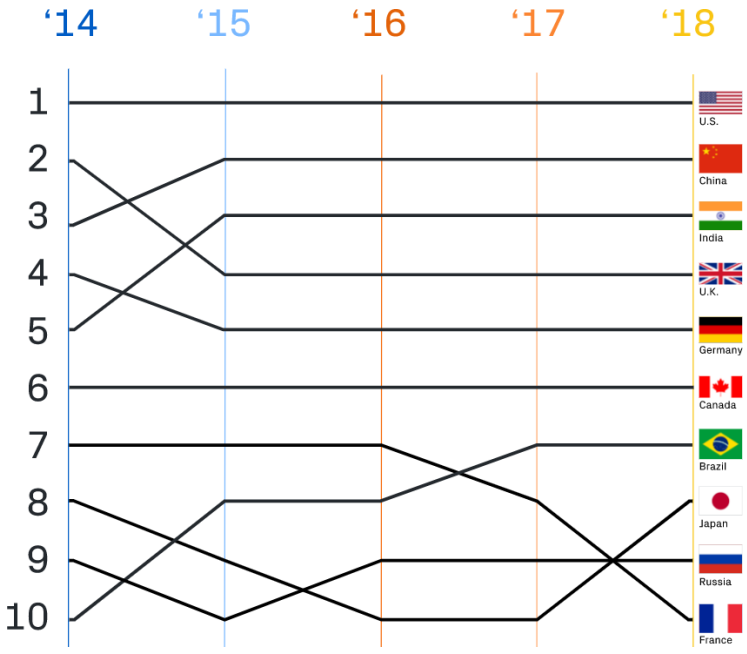

Abbildung 4.7: Standorte mit den meisten GitHub-Benutzer [3]

4.1.2 Statistiken über gefundene BPMN-Prozesse

Alle gefundenen Dateien wurden in zwei disjunkte Gruppen geteilt: BPMN-Prozesse, gespeicherte im XSD-Serialisierungsformat und alle anderen Dateien, die

in ihrem Namen das Wort „bpmn“ enthalten. Beide diese Gruppen wurden an solche Merkmale analysiert, wie Anzahl der Autoren, Dateialter und Anzahl der Dateiänderungen.

4.1.2.1 Anzahl der Autoren

Die Abbildungen 2.8 und 2.9 zeigen uns Anzahl der Autoren von Dateien aus beiden Gruppen. Aus diesen Diagrammen kann man ablesen, dass die absolute Mehrheit von Dateien (91.3% der BPMN-Prozesse im XSD-Serialisierungsformat und 95,2% der anderen Dateien) nur einen Autor haben. Bei BPMN-Prozessen im XSD-Serialisierungsformat ist dieser Anteil jedoch um 4,1% kleiner als bei allen anderen gefundenen Dateien. 5% der BPMN-Prozesse im XSD-Serialisierungsformat haben 2 Autoren. Bei allen anderen Dateien ist dieser Anteil niedriger und beträgt 3.2%.

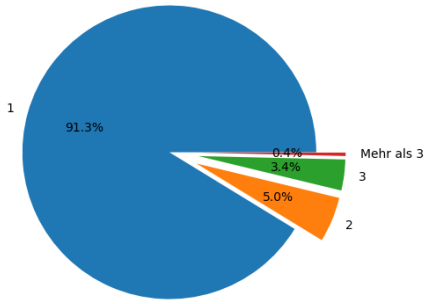

Abbildung 4.8: Anzahl der Autoren von BPMN-Prozessen im XSD-Serialisierungsformat

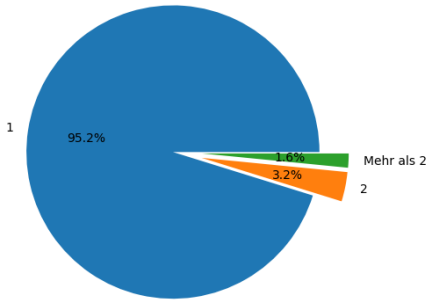

Abbildung 4.9: Anzahl der Autoren von Dateien ohne BPMN-Prozesse im XSD-Serialisierungsformat

Insgesamt zeigen uns Abbildungen 2.8 und 2.9, dass die absolute Mehrheit von allen gefundenen Dateien einen Autor haben. Signifikant ist auch, dass Anzahl der Dateien mit 2 und 3 Autoren bei BPMN-Prozesse im XSD-Serialisierungsformat größer ist als bei allen anderen Dateien.

4.1.2.2 Dateialter

Die Abbildungen 2.10 und 2.11 zeigen uns das Alter von Dateien aus beiden Gruppen. Es ist zu sehen, dass die Daten auf beiden Grafiken ähnlich sind. Die Mehrheit der BPMN-Prozesse im XSD-Serialisierungsformat (38,9%) und Mehrheit von allen anderen Dateien (36,1%) sind nicht älter als 1 Jahr. Fast jede vierte Datei auf beiden Diagrammen ist 1 Jahr alt und jede sechste Datei - 2 Jahre alt. Die beiden Diagrammen unterscheiden sich nur in dem, dass die ältesten BPMN-Prozesse im XSD-Serialisierungsformat vor 8 Jahren erstellt wurden und alle anderen Dateien - vor 11 Jahren.

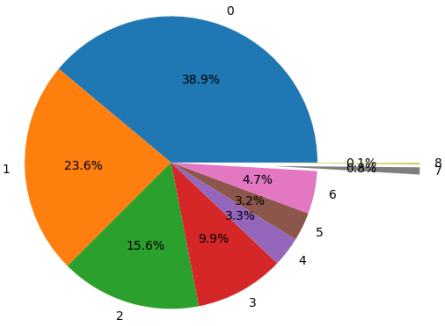

Abbildung 4.10: Alter der BPMN-Prozesse im XSD-Serialisierungsformat

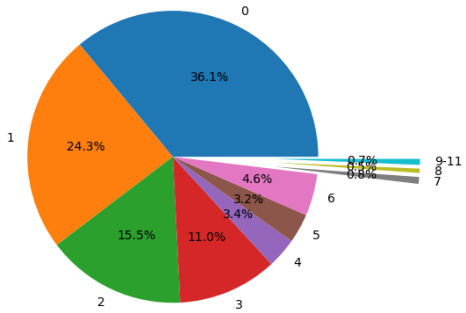

Abbildung 4.11: Alter der Dateien ohne BPMN-Prozesse im XSD-Serialisierungsformat

4.1.2.3 Anzahl der Dateiänderungen

Das nächste Merkmal von Dateien aus beiden Gruppen, das untersucht wurde, war die Anzahl der Dateiänderungen inklusive der Dateierstellung. Wie aus der Abbildungen 2.12 und 2.13 zu entnehmen ist, gibt es einige Unterschiede zwischen den beiden Gruppen von Dateien. So wurden 90,1% von allen Dateien ohne BPMN-Prozesse im XSD-Serialisierungsformat nach der Erstellung nicht mehr geändert. Bei BPMN-Prozesse im XSD-Serialisierungsformat ist der Anteil von solchen Dateien um 17,1% niedriger und beträgt 73%. Der Anteil der Dateien die zweimal geändert wurden ist noch unterschiedlicher und beträgt bei BPMN-Prozess im XSD-Serialisierungsformat 18,9% und bei allen anderen Dateien nur 4,2%.

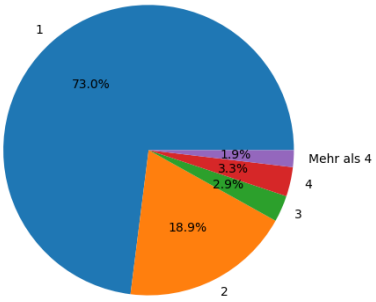

Abbildung 4.12: Anzahl der Änderungen von BPMN-Prozessen im XSD-Serialisierungsformat

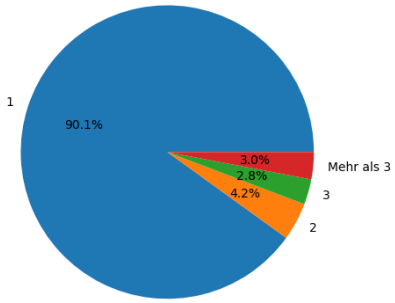

Abbildung 4.13: Anzahl der Änderungen von gefundenen Dateien ohne BPMN-Prozesse im XSD-Serialisierungsformat

4.1.2.4 Analyse der Dateien bezüglich Duplikate

Auf den Abbildungen 2.12 und 2.13 sind Informationen über Duplikate innerhalb der beiden Gruppen von Dateien dargestellt. Es ist zu sehen, dass nur für 40% der BPMN-Prozesse im XSD-Serialisierungsformat und 28,4% der anderen Deteien keine Duplikate gefunden wurden. Die Anteile der Dateien, die mindestens einmal dupliziert wurden, sind auf beiden Grafiken ähnlich: 12,6% und 14,3%. Aber der Anteil der reinen Duplikate ist bei BPMN-Prozesse im XSD-Serialisierungsformat um 10% kleiner als bei anderen Dateien und beträgt 47,3%.

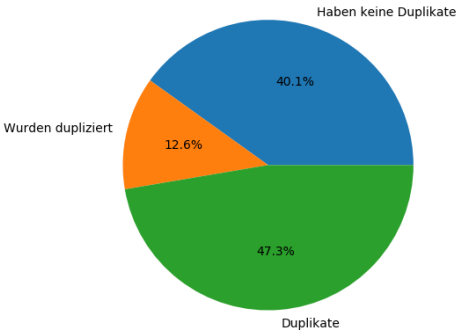

Abbildung 4.14: Angaben über Duplikate von BPMN-Prozessen im XSD-Serialisierungsformat

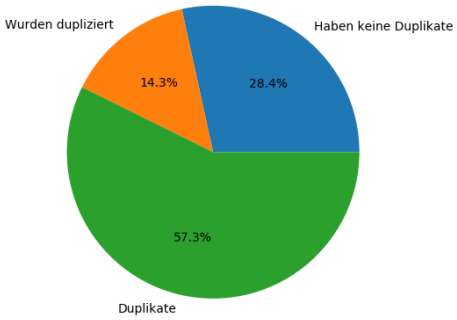

Abbildung 4.15: Angaben über Duplikate von gefundenen Dateien ohne BPMN-Prozesse im XSD-Serialisierungsformat

4.2 Statistiken über Korrektheit der BPMN-Prozesse im XSD-Serialisierungsformat

Wie Matthias Geiger - Autor von BPMNspector angibt, werden BPMN-Prozesse häufig mit einigen Verstößen gegen BPMN 2.0-Einschränkungen erstellt[2]. Deshalb war es für uns von Interesse, die gefilterten BPMN-Prozesse an solche Verstöße mithilfe des BPMNspectors zu untersuchen. Dieses Programm analysiert ein oder mehrere BPMN-Prozesse und ermittelt ob jeder von ihnen valide ist oder nicht. Falls ein Prozess nicht valide ist, liefert BPMNspector alle Verstöße gegen BPMN 2.0-Einschränkungen, die er hat. Die Liste von allen möglichen Verstößen ist in der Publikation „BPMN 2.0 Process Model Serialization Constraints“ von Matthias Geiger definiert. Siehe[5].

Es wurden insgesamt 8.905 BPMN-Prozesse im XSD-Serialisierungsformat untersucht. Dazu zählten 6.779 BPMN-Prozesse, für die es keine Duplikate gefunden wurden und 2.126 BPMN-Prozesse, die dupliziert wurden. Ausgenommen aus der Untersuchung waren 8.002 reine Kopien von BPMN-Prozessen.

Wie auf der Abbildung 2.16 zu sehen ist, wurden nur 16,5% von den untersuchten BPMN-Prozessen als valide Prozesse eingestuft. In 83,5% von untersuchten BPMN-

Prozessen wurden verschiedene Verstößen gegen BPMN 2.0-Einschränkungen gefunden.

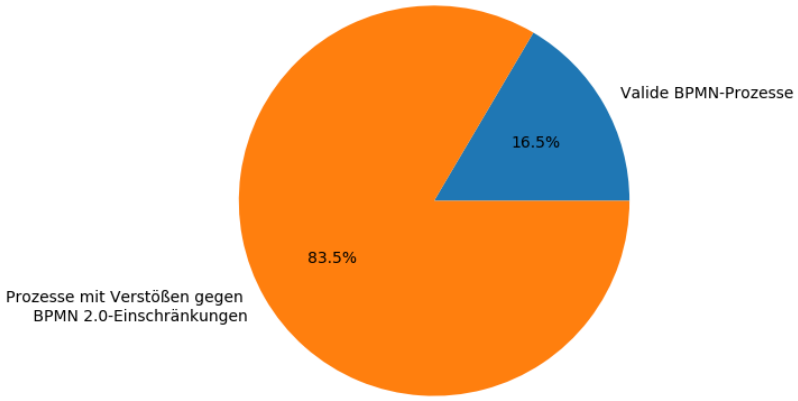

Abbildung 4.16: Prozentsatz der Verstöße gegen BPMN 2.0-Einschränkungen

Die Abbildung 2.17 zeigt uns Statistik über 10 Verstöße gegen BPMN 2.0-Einschränkungen, die in meisten nicht validen BPMN-Prozessen gefunden wurden. Es ist zu sehen, dass drei Arten der Verstöße, nämlich „EXT.101“, „EXT.023“ und „EXT.107“, in absoluter Mehrheit von nicht validen BPMN-Prozessen gefunden wurden.

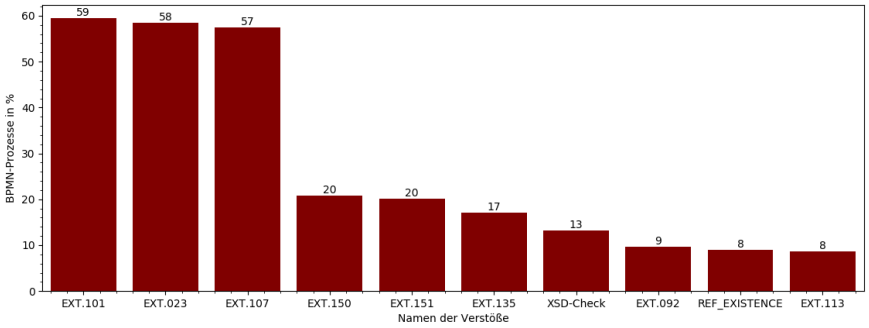

Abbildung 4.17: Top 10 Verstöße gegen BPMN 2.0-Einschränkungen,

Der „EXT.101“ Verstoß betrifft das Element Startereignis (StartEvent). Hier geht es um Regelverletzung, die besagt, dass „ein Startereignis eine Quelle für einen Sequenzfluss sein muss“[5].

Der „EXT.023“ Vertoß betrifft das Element Sequenzfluss (SequenceFlow). Es wird eine Regel verletzt, die besagt, dass „das Quell- und Zielelement des Sequenzflusses unter Verwendung der eingehenden oder ausgehenden Attribute auf die SequenceFlow-Definition verweisen muss“[5]. Der Autor des BPMNspectors angibt, dass „EXT.023“ ein der häufigsten Verstöße bei BPMN-Prozessen ist [1]. Unsere Anwendung des BPMNspectors hat diese Aussage vollständig bestätigt. Matthias Geiger bietet das von ihm entwickeltes „BPMNspector-fixSeqFlow“-Programm[1], das diese Art von Verstößen korrigiert.

Der „EXT.107“ Verstoß betrifft das Element Endereignis (EndEvent). Es geht hier um Verletzung einer Regel, die besagt, dass „ein Endereignis mindestens einen ein-

gehenden Sequenzfluss haben muss“[5].

Die Abbildung 2.18 zeigt uns Top 10 Verstöße gegen BPMN 2.0-Einschränkungen, aufsummiert bei allen BPMN-Prozessen. Es ist zu sehen, dass der Verstoß „EXT.023“ die größte Anzahl hat, nämlich 58.015. Im Unterschied zur Abbildung 2.17, wo „XSD-Check“ den 7. Platzt hat und bei 13% von nicht validen BPMN-Prozessen vorkommt, befindet er sich auf der Abbildung 2.18 mit der Anzahl 43.516 auf dem zweiten Platz. Die Anzahle der anderen Verstöße, die auf der Abbildung 2.18 zu sehen sind, sind im Vergleich zu schon erwähnten deutlich kleiner.

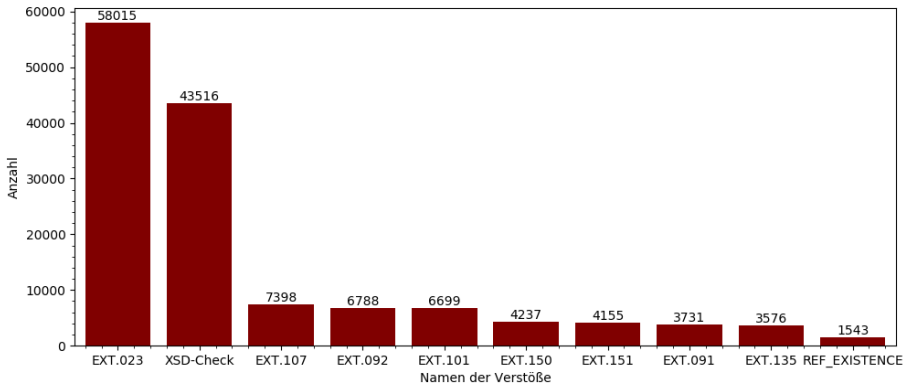

Abbildung 4.18: Top 10 Verstöße gegen BPMN 2.0-Einschränkungen (aufsummiert)

Insgesamt hat Anwendung des BPMNspectors auf die gefundene BPMN-Prozesse im XSD-Serialisierungsformat gezeigt, dass nur jeder sechste von ihnen im Hinsicht auf die BPMN-Spezifikation ein korrekter Prozess ist. Daraus kann man schließen, dass es ein Bedarf für die Verbesserung der modernen BPMN-Werkzeuge existiert. Die Grafiken 2.17 und 2.18 machen es deutlich, welche Verstöße gegen BPMN 2.0-Einschränkungen bei der Erstellung der Prozesse berücksichtigt werden müssen.

Literaturverzeichnis

[1] Bpmnspector-fixseqflow. https://github.com/matthiasgeiger/BPMNspector-fixSeqFlow. Accessed: 2019-06-11.

[2] Bpmnspector. static analysis of bpmn 2.0 process models. http://bpmnspector.org/. Accessed: 2019-05-12.

[3] Duplicate files finder. http://doubles.sourceforge.net/. Accessed: 2019-05-12.

[4] formal/13-12-09. https://www.omg.org/spec/BPMN/2.0.2. Accessed: 2019-04-15.

[5] Ghtorrent on the google cloud. http://ghtorrent.org/gcloud.html. Accessed: 2019-04-19.

[6] The ghtorrent project. http://ghtorrent.org/. Accessed: 2019-04-19.

[7] Github facts. https://github.com/about/facts. Accessed: 2019-04-19.

[8] Google bigquery. https://cloud.google.com/bigquery/. Accessed: 2019-04-19.

[9] Places with the most contributors. https://octoverse.github.com/people. Accessed: 2019-05-22.

[10] Rest api v3. https://developer.github.com/v3/. Accessed: 2019-04-22.

[11] The state of the octoverse. https://octoverse.github.com/. Accessed: 2019-04-19.

[12] The state of the octoverse: top programming languages of 2018. https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/. Accessed: 2019-05-22.

[13] Rasmus Voss Jakub Narebski Ferdinando Santacroce, Aske Olsson. *Git: Mastering Version Control*. Packt Publishing Ltd, Birmingham, 2016.

[14] Matthias Geiger. BPMN 2.0 Process Model Serialization Constraints. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik, no. 92, Otto-Friedrich Universität Bamberg, May 2013.

[15] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. Lean ghtorrent: Github data on demand. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 384–387, New York, NY, USA, 2014. ACM.

[16] Regina Hebig, Truong Ho Quang, Michel R. V. Chaudron, Gregorio Robles, and Miguel Angel Fernandez. The quest for open source projects that use uml: Mining github. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, MODELS '16, pages 173–183, New York, NY, USA, 2016. ACM.

[17] Gregorio Robles Miguel Angel Fernandez. Extracting software development information from floss projects in github. Paper at SATToSE 2017, `http://sattose.wdfiles.com/local--files/2017:schedule/SATToSE_2017_paper_6.pdf`, 2017. Accessed: 2019-04-20.

[18] Gregorio Robles. 2016-uml-miner. `https://github.com/LibreSoftTeam/2016-uml-miner`. Accessed: 2019-04-23.

[19] Gregorio Robles, Truong Ho-Quang, Regina Hebig, Michel Chaudron, and Miguel FernÃ¡ndez. An extensive dataset of uml models in github. pages 519–522, 05 2017.

[20] Gregorio Robles, Truong Ho-Quang, Regina Hebig, Michel R. V. Chaudron, and Miguel Angel Fernandez. An extensive dataset of uml models in github. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, pages 519–522, Piscataway, NJ, USA, 2017. IEEE Press.

[21] Adam Tornhill. *Your Code as a Crime Scene - Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs*. Pragmatic Bookshelf, Raleigh, Noth Caolina and Dallas, Texas, 2015.